software application or package. Examples of the descriptive information **104** can include feedback, reviews, questions, or comments about the software project; source code for the software project; configuration files or readme files provided with the software project; keywords characterizing the software project; a description of the software project; or any combination of these. The computing device **102** can obtain the descriptive information **104** from one or more sources **106** (e.g., over the Internet). Examples of the source(s) **106** can include one or more websites, such as discussion forums, repositories, review websites, or any combination of these. The source(s) **106** can also include parts of the software project **122**, such as readme files, configuration files, or source code files.

[0012] As one particular example, the computing device **102** can access a website for an online repository. The website can have a description of the software project **122** and host source code for the software project **122**. The website can also list keywords (e.g., topics or tags) characterizing the software project **122**. The keywords may have previously been input by users or automatically generated by the website. The website may further have a feedback section, through which users can post comments, reviews, questions, and bugs relating to the software project **122**. The computing device **102** can access the website to retrieve some or all of this information.

[0013] After obtaining the descriptive information **104** for the software project **122**, the computing device **102** can parse the descriptive information **104** to determine software features **110** of the software project **122**. The software features **110** can be any functional characteristics of the software project **122** relating to how the software project **122** works or operates. For example, the software features **110** can include tasks and functions that the software project **122** is configured to perform, frameworks and dependencies relied on by the software project **122**, operating systems and operating environments for the software project **122**, or any combination of these. The computing device **102** can determine the software features **110** based on the descriptive information **104** using any number and combination of techniques.

[0014] As one particular example, the descriptive information **104** may include website content scraped from a website, such as stackoverlow™. The website content can include a question relating to the software project **122**, along with keywords characterizing the software project **122**. The keywords may have previously been input by the user posing the question. In some such examples, the computing device **102** can determine the software features **110** by parsing the keywords from the website content and using at least some of those keywords as the software features **110**.

[0015] As another example, the computing device **102** can apply a count technique to the descriptive information **104** to determine the software features **110**. The count technique can involve counting how many times a particular textual term occurs in the descriptive information **104** and storing that count value. For example, if the textual term "Python" is present **32** times in the descriptive information **104**, then the count for that textual term would be **32**. The computing device **102** can iterate this process to determine counts for some or all of the textual terms in the descriptive information **104**. The computing device **102** can then determine which of the textual terms have counts exceeding a predefined threshold (e.g., 30). Those textual terms may indicate particularly important software features. So, at least some of those textual terms may be designated as the software features **110**. In some cases, a portion of the textual terms may be filtered out (e.g., using a predefined filter list) as irrelevant, for example, because they are articles, prepositions, or otherwise relatively common textual terms that provide little value, to improve accuracy.

[0016] As still another example, the computing device **102** can apply a machine-learning model **112** to the descriptive information **104** to determine the software features **110**. Examples of the machine-learning model **112** can include a deep neural network, a Naïve Bias classifier, or a support vector machine. The machine-learning model **112** can be configured to analyze textual information and identify software features **110** therein. For example, the machine-learning model **112** may include a deep neural network that is trained using training data. Each entry in the training data can include a relationship between a keyword and a flag indicating whether or not the keyword relates to a software feature. The machine-learning model **112** can learn these relationships and then able to reliably predict whether or not an unknown keyword relates to a software feature.

[0017] The computing device **102** can apply any number and combination of techniques discussed above to determine the software features **110** of the software project **122**. The computing device **102** can additionally or alternatively apply other techniques, such as term frequency-inverse document frequency ("TF-IDF"), to determine the software features **110** of the software project **122**. TF-IDF may involve generating a numerical statistic that reflects how important a word is to a document in a corpus. In the present context, the document may be a file or website associated with the software project **122** and the corpus is the descriptive information **104**. The TF-IDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.

[0018] Having determined the software features **110** of the software project **122**, the computing device **102** can next determine a feature vector **114** for the software project **122**. In some examples, the computing device **102** can determine the feature vector **114** for the software project **122** by first generating a default feature vector in which all the elements have default values (e.g., zeros). One simplified example of the default feature vector can be {0, 0, 0, 0, 0, 0}. Each element in the default feature vector can be mapped to a particular software feature. For example, the elements in the above default feature vector can be mapped to the following: {C++, Openshift, Tensorflow, Linux, Machine-learning, Python}. If the computing device **102** determined that the software project **122** has a particular software feature, the computing device **102** can then modify the corresponding element's value in the default feature vector to so indicate. For example, if the computing device **102** determined that software project **122** relies on "tensorflow" (an open source machine-learning library for research and production), then the computing device **102** can change the third element's value to one, yielding a feature vector of {0, 0, 1, 0, 0, 0}. The computing device **102** can repeat this process for each of the software features **110** to generate the feature vector **114** for the software project **122**.

[0019] After generating the feature vector **114** for the software project **122**, the computing device **102** can store the